Logical partition mode physical resource management on the IBM eServer z990

I. G. Siegel
B. A. Glendening
J. P. Kubala

The IBM eServer™ z990 provides tremendously increased processor, I/O, and memory capacity exceeding the capability of even the premier IBM operating systems. The modular or book-form system topology of the z990 enables a highly flexible and more cost-effective concurrent upgrade infrastructure, as well as improved hardware failure survivability and serviceability. The multibook form of the z990 has two significant memory access performance issues which are addressed here: First, there is increased cache coherency overhead when the same memory is accessed by central processing units (CPUs) from multiple books; second, access from CPUs to memory on books other than the book on which a CPU is resident is not as efficient as access from the same book. Awareness of this multifold increase in capacity and complexity is effectively managed by the IBM zSeries® logical partition (LPAR) hypervisor, obviating the need for operating system involvement. This paper describes changes made to the zSeries LPAR hypervisor to manage CPU and memory resources on the z990 machine topology.

Introduction

Traditional zSeries* symmetric multiprocessors (SMPs) have had cache-coherent, tightly coupled processors that have scaled up from uniprocessor models to 16-way models [1]. IBM zSeries servers can be partitioned into separate logical computing systems. System resources (memory, processors, I/O devices) can be divided or shared among many such independent logical partitions (LPARs) under the control of the LPAR hypervisor,[1] which comes with the standard Processor Resource/Systems Manager* (PR/SM*) feature on all zSeries servers [2]. Each LPAR supports an independent operating system (OS) loaded by a separate initial program load (IPL) operation.

Logical partitions contain one or more logical central processing units (CPUs), which can be defined to be dedicated or shared. The LPAR hypervisor dispatches a dedicated logical CPU exclusively on a chosen physical CPU. Shared logical CPUs are dispatched on any remaining physical CPUs, chosen not for dedication but in accordance with a user-assigned priority expressed in terms of a relative weight [3]. Most users define the logical CPUs to be shared. This allows the LPAR hypervisor to maximize the use of the available physical CPUs depending on the activity and weight of the logical CPUs.

The zSeries z990 models offer a platform for large workload consolidations. Comprising one to four books, each with up to eight processing units (CPUs) and a shared Level 2 (L2) cache, a z990 configuration can have as many as 32 CPUs. Each book can have memory attached to it which is accessible from all of the books. The accelerating costs of building the traditional tightly coupled interconnections between CPUs become prohibitive as the number of processors is increased.

[1] The hypervisor is a software layer to manage multiple operating systems running in a single central processing complex.

0018-8646/04/$5.00 © 2004 IBM

A modular system topology such as that of the z990 helps to alleviate these costs. By using PR/SM, such a system can be broken up in logically partitioned mode into as many as 30 logical partitions, all under the control of the LPAR hypervisor.

The multibook form of the z990 has two significant memory access performance issues that must be addressed:

1. There is an increased cache coherency overhead when the same memory is accessed by CPUs from multiple books.
2. Access from CPUs to memory on books other than the book on which a CPU is resident is not as efficient as that from the same book.

These topology issues, whose performance impact was not significant in previous server designs, provide opportunities for the LPAR hypervisor to optimize allocation of resources and shield operating systems from an awareness of the underlying server topology.

To curtail these effects, the LPAR hypervisor must minimize multiple-book access of the same piece of memory and off-book memory accesses. A given range of memory is owned (accessible) by only one logical partition, so multiple-book access can be minimized by having the logical CPUs of the owning logical partition dispatched, as much as possible, on only a single book. Additionally, off-book access is dramatically reduced by preferential dispatch of the logical CPUs of a logical partition on the same book as that from which memory was allocated. This paper discusses how the LPAR hypervisor realizes these goals by establishing a book affinity for each of the logical CPUs of a logical partition and optimal memory allocation with regard to the established logical CPU book affinities.

Opportunity for enhancement

The zSeries platform continues to grow and provide for consolidation of multiple workloads onto a single platform. Workloads running in multiple logical partitions under PR/SM can take advantage of unused capacity on the SMP. The z990 can provide up to 32 processors for customer use on a single SMP; this is done with up to eight processors on a tightly coupled multiple-chip module (MCM) which, by itself, resembles a traditional zSeries SMP. Each MCM has a 32-MB shared L2 cache plus I/O connections. This L2 cache is a processor cache, not a memory cache; its contents reflect the most recently accessed portions of memory from the processors on that MCM. Portions of this MCM memory that are referenced only by processors on other MCMs are stored only in the L2 caches of the other MCMs.

For the purposes of this paper, an MCM along with its attached memory is referred to as a book. Up to four such books can be configured in a fully cache-coherent, single-SMP configuration, with a ring interconnect between the books. Performance can be optimized by limiting reference (update) of a given piece of memory by multiple books. Also, additional latencies and bandwidth constraints exist when a CPU accesses memory on a nonlocal book. This is understood by applications that were designed for a nonuniform memory access (NUMA) environment, but the idea is largely foreign to traditional zSeries operating systems and applications that have grown up on SMPs. This paper describes how PR/SM manages CPU and memory resources while shielding the traditional operating systems from these issues.

Implementation

From its inception, the z990 system was designed to run always in LPAR mode. This decision makes it possible to exploit the fact that the LPAR hypervisor is always present and it can be utilized to conceal changes to the underlying system infrastructure from user operating systems and applications. This was a natural progression, since the great majority of zSeries users already operated their systems in LPAR mode rather than as a single operating system in basic (non-LPAR) mode, even when the user was running only one instance of an operating system on an eServer*. This decision allows the system to be a truly PR/SM-managed multiprocessor.

This paper focuses on the way in which PR/SM was changed to understand the underlying system topology of the z990 system and how it makes decisions in its allocation of CPU and memory resources for logical partitions.

Book topology of the z990

At present, each book of the z990 contains up to eight processing units (CPUs) that are available for use, as well as some amount of installed memory. **Figure 1** shows a conceptual example of a four-book system with 32 CPUs and 256 GB of memory. The books are numbered from 0 to 3, with the CPUs on each book using the book number as the first hexadecimal digit in their assigned CPU addresses. Each CPU contains two dedicated Level 1 (L1) caches, one for instructions and one for operands. All of the CPUs on a book share a 32-MB L2 cache. Attached to each book is 64 GB of memory, which is conceptually accessed through the local L2 cache for a total of 256 GB of system memory. The L2 caches are connected with a bidirectional ring interconnect to allow any L2 cache to access any memory through L2-to-L2 communication.

The following section describes how the book-form topology is examined in determining where the book assignments of logical CPUs and memory should be made.

Allocating resources to logical partitions on the z990

Each book of a z990 system has its own L2 memory cache, shared by the processors on that book. Access to local L2 is relatively rapid; however, as storage is shared by CPUs on different books, interbook cache communication is required in order to maintain coherency of the data, which takes significantly more time. Updating such storage can be particularly time-consuming, since more steps are required to maintain coherency. Optimal system performance can be achieved by minimizing the amount of data sharing by CPUs on different books. Also, access to memory is faster from a CPU on the book on which the memory resides than from a CPU of a different book.

The LPAR hypervisor attempts to minimize multiple-book accesses of the same piece of memory and off-book memory accesses. Since a given range of memory is owned (accessible) by only one logical partition, multiple-book access can be minimized by having the logical CPUs of the owning logical partition dispatched, as much as possible, on only one book.

A logical partition is provided access to a contiguous range of absolute addressable memory. Translation of absolute addressable memory to physical memory is provided via a configuration array in the hardware system area (HSA) in units of 64 MB, each of which is known as a memory increment. The allocation of physical memory to a logical partition and the establishment of physical CPU affinity for its logical CPUs should result in a configuration in which cross-book cache interrogation and multiple-book access to a given piece of memory are minimized. A given range of addressable memory increments is owned by only one logical partition, so performance is optimized by dispatching the logical CPUs of the owning logical partition on as few books as possible. Though this *could* be achieved by limiting dispatch of the logical CPUs of a shared logical partition to one "preferred" book, we would lose the ability to dispatch all of them simultaneously if the number of available physical CPUs on the book was less than the number of online logical CPUs.

Historically, allocations of memory resources and CPU resources for logical partitions have been completely independent. The structure of the z990 system creates the opportunity for these allocations to consider the intersection of both resource types. This section describes the changes made in support of this opportunity.

For the purposes of the following discussion, a "solution" (be it for CPUs or for memory) is a set of one or more books which provide all of the physical resources required by a logical partition. A solution for memory can differ from that for the logical CPUs of a logical partition, but each influences the choice of the other. Because the L2 cache is accessed more frequently than memory, the logical CPU solution has a more significant impact on performance than the memory solution for a logical partition. Ideally, all of the physical memory for a logical partition is allocated from one book, and the logical CPU affinities for that logical partition are established on the same book.

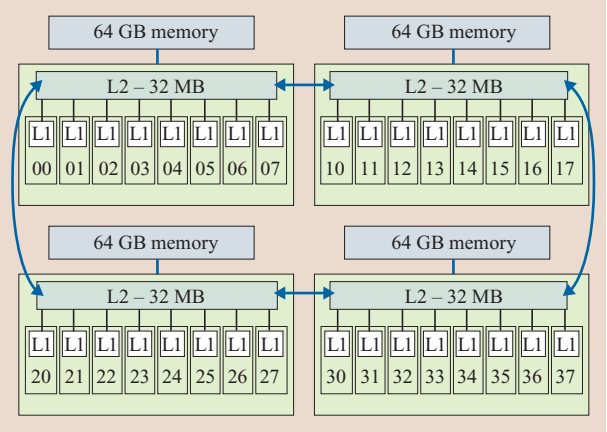

Figure 1

Conceptual example of a four-book, 32-CPU system with 256 GB of memory.

To achieve this ideal, there are several different goals of varying importance to consider when evaluating potential solutions. The following goals are applied to the allocation of resources, listed from highest to lowest priority:

1. Logical CPUs should be distributed as evenly as possible across the available physical CPUs so that maximum utilization of these physical CPUs is achieved.
2. Logical CPUs with the same logical partition should be concentrated on the smallest number of books possible in order to minimize cross-book cache interrogation overhead.
3. The degree to which the logical CPUs and memory of the logical partition are on the same book should be maximized.
4. A "best-fit" solution is preferred to a "most available" solution. A best-fit solution is one with the smallest amount of resource (memory on a book, for instance) in excess of what is required. A solution with the greatest amount of resource in excess of what is required is said to have the most available. Consistent choice of a best-fit solution over most available provides the greatest likelihood that subsequent requests will find enough resource available.

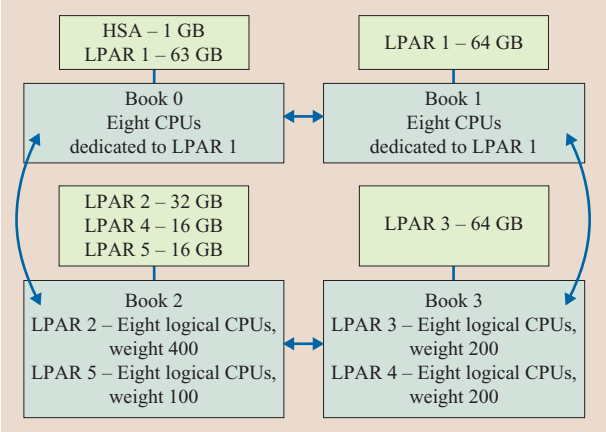

Figure 2

Resource allocations for the sample logical partition activation sequence for the system described in Figure 1.

For each logical partition, each potential solution is identified and evaluated with respect to the degree to which it satisfies all of the goals. The most highly valued solution is then implemented. For memory solutions, this means assigning memory increments from the chosen book(s) to back the absolute memory increments assigned to the activating logical partition. Logical CPUs to which physical CPUs are dedicated have a hard affinity for those CPUs because they are always dispatched on only the physical CPUs dedicated to them. Shared logical CPUs have a soft affinity for the nondedicated physical CPUs on whatever book has been chosen. If they cannot be dispatched on those physical CPUs at any particular time, others in the system on a nonpreferred book are considered. Since the evaluation process is done sequentially for each logical partition, it is possible that after the best solution has been determined for a particular logical partition (especially one with dedicated logical CPUs), it might be possible to improve upon the solutions for previous logical partitions. This reevaluation is a necessary step toward obtaining the best overall solutions for all of the logical partitions, and is discussed in more detail below.

Figure 2 illustrates the expected results on a system as described in Figure 1 (with 1 GB of HSA on Book 0) with the following logical partition definitions, activated in the order shown:

1. LPAR 1 – sixteen dedicated logical CPUs and 127 GB of memory.
2. LPAR 2 – eight shared logical CPUs with a weight of 400, and 32 GB of memory.
3. LPAR 3 – eight shared logical CPUs with a weight of 200, and 64 GB of memory.
4. LPAR 4 – eight shared logical CPUs with a weight of 200, and 16 GB of memory.
5. LPAR 5 – eight shared logical CPUs with a weight of 100, and 16 GB of memory.

Note: For the logical CPUs of LPAR 4, affinity for the physical CPUs of Book 3 will be established even though memory from Book 2 will be allocated, because at the time of activation, the physical CPUs of Book 3 are relatively underutilized, and it is more important to optimize logical CPU affinity than to optimize memory affinity.

In the example of Figure 2, when LPAR 1 is activated, any combination of two books have sufficient resources to satisfy the LPAR 1 CPU and memory requests. Each pair of books is evaluated according to the principles outlined above. For example, a solution made up of Book 0 and Book 1 has a solution value assigned to it that incorporates the following: It provides a "best fit" for the requirements (no CPUs are available in excess of those required); no other LPARs would be adversely affected, resulting in overcommitment of their shared logical CPUs to the remaining resources; all of the memory for LPAR 1 is allocatable on books 0 and 1; the concentration of logical CPUs on each book is maximized; and no other workload is currently running on the CPUs in those books.

Each of the other possible pairs of books (for instance, books 2 and 3) evaluate identically from a CPU perspective. Allocation of memory is used to differentiate between the possible pairs of books. LPAR 1 requires 127 GB of storage, which is one whole book (64 GB) plus 63 GB. The best "solution" for memory must utilize book 0 because it has only 63 GB of memory available for allocation and so provides a best fit when combined with, for instance, book 1. Any other book in conjunction with book 0 provides a best fit as well. By evaluating pairs of books in the order 0 and 1, 0 and 2, 0 and 3, etc., the book pair 0 and 1 is "discovered" first and thus is chosen as the best possible solution.

With this solution, eight logical CPUs for LPAR 1 are assigned a home book of 0 and are dedicated to physical CPUs on book 0. The other eight logical CPUs for LPAR 1 are assigned a home book of 1 dedicated to physical CPUs on book 1. Similarly, the eight shared logical CPUs for LPAR 2 are assigned a home book of 2 and are preferentially dispatched on the physical CPUs on book 2. The next section describes how this preferential dispatching is done.

Dispatching decisions on the z990

Physical CPU selection by the LPAR hypervisor dispatcher is controlled by affinity masks. Each logical CPU has an affinity mask, which represents the candidate physical CPUs on which the logical CPU can run. This mask is

called the *global affinity mask*, since it represents the complete set of physical CPUs on which the logical CPU can run. These masks take into account any asymmetrical features a processor might have, physical CPUs that have been dedicated to logical CPUs, and special types of CPUs (such as integrated coupling facilities, or ICFs). For the z990, as stated above, each logical CPU has been assigned a primary home book. In addition, a second, primary book affinity mask has been established for each logical CPU with the candidate physical CPUs on the assigned home book. Both the global and the primary book affinity masks are created as a result of the preceding resource allocation algorithms. For a dedicated logical CPU, the global and the primary book affinity masks are equal and contain only one physical CPU. For a shared logical CPU, such as one from LP2, the primary book affinity mask contains all of the physical CPUs from book 2 and the global affinity mask contains all of the physical CPUs from books 2 and 3. The dispatcher selection process is then modified to utilize this primary book information.

The first choice for assigning a logical CPU is to an idle physical CPU (i.e., a physical CPU in wait state). The available physical CPU must be a valid candidate for the logical processor. Prior to the advent of the z990 system, the hypervisor first sought to match the global affinity mask of the logical processor with the idle physical CPUs. If the physical CPU on which this logical CPU was last dispatched was currently available *and* no other logical CPU had since been dispatched on that physical CPU, the logical CPU was assigned to that same physical CPU.

The first change to the assignment algorithms for the z990 system is that the check for any other logical CPU being dispatched on this physical CPU is removed. In prior machines, if some other logical CPU had run on the physical CPU in the interim, no L1 cache or translation-lookaside buffer (TLB) entries for this logical CPU would remain in the physical CPU. The new z990 second-level translation-lookaside buffer (TLB-2) makes reassignment to the same physical CPU in this situation very fortuitous. Multiple sets of TLB-2 entries are "cached" and managed for each physical CPU.

If it is determined that the last physical CPU on which the logical CPU was dispatched is *not* available, a search is made of the remaining idle candidates for the least recently dispatched physical CPU. The least recently dispatched algorithm is used in an attempt to choose a physical CPU that is least likely to be a "good choice" for some other logical CPU that may soon request assignment. During the search, physical CPUs that were last dispatched for the same logical partition as the requesting logical CPU are biased to appear better because some useful data may exist in the L2 cache for the common data from the same logical partition.

The second change to the assignment algorithms for the z990 system is in this least recently dispatched search. In the event that a logical CPU has migrated away from its home book because of previous conditions and contention, where should it be dispatched next? Is the best approach to immediately try to bring home that logical CPU, or to attempt to make use of the residual L2 cache on the book to which the logical CPU migrated? The notion that the migration would have taken place because of contention on the home book may, in itself, indicate that the home book was overallocated for the actual assigned workload(s). Initially, it appeared that the best course of action to obtain a performance benefit would be to attempt to find a physical CPU on the book to which the logical CPU was last dispatched, even if this is not the home book of the logical CPU. This would dynamically make use of the resources to which the logical CPU was last dispatched. However, performance tests showed that the most important criterion was to return to the home book as quickly as possible. This criterion was shown to be important enough to override even the match with the last dispatched physical CPU as well.

Thus, the net change for z990 is that the search for the least recently dispatched idle physical CPU was modified to search one of the following three lists (The first nonzero list is the only list that will be searched since, if nonzero, a place to assign the logical CPU is guaranteed.):

1. A list of idle candidate physical CPUs on the home book of the logical CPU.
2. A list of idle candidate physical CPUs on the same book to which this logical CPU was last dispatched.
3. A list of all idle candidate physical CPUs on the machine.

As seen above, if there is an idle physical CPU available for selection, the logical CPU is dispatched immediately to a physical CPU. There is no notion of delaying a logical CPU dispatch because a physical CPU on its home book is not currently available.

Next, the dispatcher must deal with situations in which no idle candidate physical CPUs are available for selection. Basically, if no physical CPUs are idle, the lowest-priority dispatched unit of work (logical CPU) on a candidate physical CPU is sought. The displaced unit of work must be of a lower priority than the unit of work being dispatched.

The search for the lowest-priority dispatched unit of work is modified for the z990 system in a manner similar to the search for idle physical CPUs. The difference here is that each of the three lists must potentially be searched, in the order stated below. The first search that finds a unit of work to displace is used. Of course, checks are made to determine that the lists are actually different in order to

avoid unnecessary repeated processing. The search order is as follows:

1. A list of candidate physical CPUs on the home book of the logical CPU (the affinity mask of the logical CPU home book).
2. A list of candidate physical CPUs on the book to which the logical CPU was last dispatched.
3. A list of all candidate physical CPUs on the system (the global affinity mask of the logical CPU).

If lower-priority work cannot be found to preempt, the logical CPU must wait its turn in the dispatch queue.

Reoptimizing primary book assignments on the z990 system

Once established, physical CPU affinity for a dedicated logical CPU cannot change because the hypervisor does not yet have the ability to rededicate a different physical CPU transparent to the logical partition.

Shared logical CPU affinities determined to be optimal at logical partition activation may require reevaluation when the availability and usage of system resources change. Since the distribution of logical CPU weights among other logical partitions can affect the optimum affinity for any particular logical partition, an occurrence necessitating reoptimization generally affects all currently active logical partitions using shared logical CPUs.

The following events require reoptimization:

- Configure ON of additional physical CPUs, as for concurrent capacity backup (CBU).
- Configure OFF of physical CPUs, as for concurrent CBU undo.
- Dedication of a physical CPU to a logical CPU if the dedication reduces the number of nondedicated physical CPUs remaining on its book below the maximum number of logical CPUs with affinity to that book, for any logical partition with shared logical CPUs.
- Checkstop[2] of a physical CPU when no spare is available on the system.
- Transparent sparing[3] of a checkstopped physical CPU when the CPU brought into the configuration comes from a book different from the one containing the broken CPU.
- Logical partition activation of shared logical partitions so that distribution of shared weights can be equalized among all of the books.
- Configure ON of a shared logical CPU. Reoptimization is limited to the logical CPUs of the initiating logical partition to avoid excessive churning of remaining logical partitions.

Future potential enhancements

Although PR/SM makes the best choice possible when allocating resources to a logical partition, there are times when a less than optimal solution is all that can be achieved, primarily because of resource fragmentation. LPAR does not currently have the infrastructure to transparently change the backing physical CPU for a dedicated logical CPU, nor is it able to dynamically change the book(s) providing backing physical increments for memory ranges owned by logical partitions. Though not as important to overall performance as logical CPU dispatching, the location of memory for a logical partition does play a role. While the current memory allocation is static, it may be possible to reallocate memory increments dynamically to better contain resource fragmentation from multiple logical partition activity.

Future eServer plans striving for even higher overall availability could include such functions as concurrent book replacement, which would require that these capabilities be supported to avoid negative impact on any logical partitions currently using the resources on the book to be replaced.

These capabilities can also be used to maintain an optimal distribution of physical CPU resources and memory among the active logical partitions, effectively eliminating compromises brought on by resource fragmentation. Further improvements could be made to evaluate the logical partitions on the basis of properties such as quantity and relative weight of their logical CPUs, and to provide the optimum solution for the most highly valued logical partitions. The current implementation is somewhat constrained by the order of logical partition activation, which does not necessarily indicate the relative importance of the workloads of the logical partitions.

Maintaining a single-system image on the z990

From the earliest design discussions for the z990 system, a primary goal has been to shield the book structure from computer operators and operating systems. An understanding of the performance and RAS (reliability, availability, serviceability) ramifications of a multiple-book system would have involved a steep learning curve and pervasive changes to all of the premier IBM operating systems. Keeping customer management of this additional complexity simple is a primary goal on the latest zSeries line of servers.

[2] In certain situations, it is impossible or undesirable to continue operation when a machine error occurs. In these cases, the CPU may enter the checkstop state. In general, the CPU may enter the checkstop state whenever an uncorrectable error or other malfunction occurs and the machine is unable to recognize a specific machine-check-interruption condition.

[3] Transparent CPU sparing is a function that automatically replaces a failed physical CPU with an unused "spare" CPU without loss of data and continues running the instruction stream with no discernible interruption. This replacement is "transparent" to the zSeries LPAR hypervisor and to any software running in a logical partition.

An IBM eServer, though comprising multiple books, each with its own CPUs, L2 cache, memory, and I/O, is nevertheless a single system. The purpose of the PR/SM has been to foster this (correct) perception by limiting multiple-book awareness and management of the disparate resources to itself. Effectively presenting a single-system image to all other entities maximizes performance capabilities and simplifies management of the z990. Having PR/SM optimize the usage of resources transparently as resources are added or removed or workloads are added or removed enhances the self-optimizing strengths of zSeries servers. Furthermore, should changes to the book-form topology be required for future systems, PR/SM can change with the system, while software in the field does not have to be updated.

Conclusions

In this paper we have described the enhanced self-managing aspects of the z990 system as they pertain to hardware resources needed to operate servers, in particular CPU and memory resources. The omnipresence of the LPAR hypervisor permits the supported number of resources to be scaled up immensely, allowing for large consolidations of servers onto a single platform. This horizontal growth does not require changes to the applications deployed on these servers, since the underlying system topology is transparent to them. The approach also allows operating systems to scale themselves up on their own timetable if and when this is required.

Acknowledgments

The authors thank the anonymous reviewers of this paper for their insightful comments.

*Trademark or registered trademark of International Business Machines Corporation.

References

1. IBM Corporation, *z/Architecture Principles of Operation* (SA22-7832); see *http://www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi/*.
2. IBM Corporation, *zSeries 990 Processor Resource/Systems Manager Planning Guide* (SB10-7036); see *http://www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi/*.
3. W. J. Rooney, J. P. Kubala, J. Maergner, and P. B. Yocom, "Intelligent Resource Director," *IBM J. Res. & Dev.* **46,** No. 4/5, 567–586 (July/September 2002).

Received September 22, 2003; accepted for publication March 18, 2004; Internet publication May 6, 2004

Ira G. Siegel *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (isiegel@us.ibm.com).* Mr. Siegel is a Staff Software Engineer supporting development in the zSeries LPAR hypervisor. He received a B.S. degree in zoology from Rutgers University and an M.S. degree in computer science from Villanova University. Since joining IBM in 1989, he has worked on LPARs, developing such diverse items as dynamic storage reconfiguration, concurrent patch, and concurrent CPU upgrade.

Beth A. Glendening *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (glenden@us.ibm.com).* Ms. Glendening is a Senior Programmer in the z/OS Core Technical Design Team Department. She received a B.S. degree in chemical engineering from Rensselaer Polytechnic Institute and an M.S. degree in computer science from Union College, joining IBM in 1982. Since then, she has worked on S/390 architecture verification test tools, z/OS workload management development, z/OS RACF development, porting Tivoli applications to the z/OS platform, and zSeries logical partitioning.

Jeffrey P. Kubala *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (kubes@us.ibm.com).* Mr. Kubala is a Senior Technical Staff Member in the z/OS Core Technical Design Team Department; he is currently the technical team leader for the zSeries LPAR hypervisor. He received a B.S.E. degree in computer engineering from the University of Connecticut, joining IBM in 1981. Since then, he has worked on compiler design and development, OS/390 Hiperbatch, and S/390 and zSeries logical partitioning. In addition to his role as the zSeries LPAR hypervisor technical team leader, he is actively engaged with the iSeries and pSeries hypervisor teams as a technical consultant.